Human Interfaces for Cooperative Control of Multiple Vehicle Systems

Jisang Sun
Brigham Young University - Provo

Follow this and additional works at: https://scholarsarchive.byu.edu/etd
Part of the Mechanical Engineering Commons

BYU ScholarsArchive Citation
Sun, Jisang, "Human Interfaces for Cooperative Control of Multiple Vehicle Systems" (2006). All Theses and Dissertations. 373.
https://scholarsarchive.byu.edu/etd/373

This Thesis is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in All Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.
This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date

Timothy W. McLain, Chair

Date

Robert H. Todd

Date

Michael A. Goodrich
As chair of the candidate’s graduate committee, I have read the thesis of Ji-Sang Sun in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date

Timothy W. McLain
Chair, Graduate Committee

Accepted for the Department

Matthew R. Jones
Graduate Coordinator

Accepted for the College

Alan R. Parkinson
Dean, Ira A. Fulton College of Engineering and Technology
ABSTRACT

HUMAN INTERFACES FOR COOPERATIVE CONTROL OF
MULTIPLE VEHICLE SYSTEMS

Ji-Sang Sun

Department of Mechanical Engineering
Master of Science

This thesis presents a human interface which helps users efficiently allocate multiple unmanned ground vehicles (UGVs) cooperating to accomplish timing-sensitive missions in an urban environment. The urban environment consists of obstacles and a hazardous region. The obstacles represent a “no-go zone” while the hazardous region represents a high-risk area. The main object of this problem is to minimize the team operational cost while satisfying timing constraints. Operational costs for individual vehicles are based on risk and power consumption, and are calculated using path length and vehicle velocity. In this thesis, three types of timing constraints are considered: simultaneous arrival, tight sequential arrival, and loose sequential arrival. Coordination variables and functions are the strategy by which both temporal and spatial information is used to achieve cooperative timing at a minimum cost. Specifically, coordination variables and functions are used to plan trajectories for a team of UGVs that satisfy timing constraints. The importance of properly representing information to users, allowing them to make efficient decisions, is also discussed. Four different control interfaces (temporal, spatial, cost, and coordination variable/
function control) were tested. A full factorial design of experiments was performed with response time, workload, and quality of decision as metrics used to evaluate the quality and effectiveness of each interface. Based on the results of this experiment, a final graphical user interface (GUI) was designed and is described. It incorporates a combination of coordination variable/function control and cost control. This GUI is capable of planning paths for vehicles based on cooperative timing constraints and enables users to make high quality decisions in deploying a group of vehicles.
ACKNOWLEDGMENTS

My gratitude is beyond description. I express my great gratitude to all who help me finish my work. First and foremost, I express my sincere thanks to my lovely wife, MinJung, for her love, support, patience, trust, and encouragement. I also sincerely express my appreciation to Dr. McLain for his support, guidance, and trust even during the hardest time of my academic career. I also thank my graduate committee members, Dr. Todd and Dr. Goodrich for their supporting, guidance, and encouragement. In addition, I thank Jeff Anderson, my lifelong friend, for his friendship and support. Furthermore, I thank Wei Ren, Derek Kingston, and Matt Blake for their time and effort. Last but not least, I am grateful to all my family members for their support to accomplish my educational achievement.
Contents

Acknowledgments vii
List of Tables xiii
List of Figures xvi

1 Introduction 1
   1.1 Problem Statement 3
       1.1.1 Cooperative Timing Problem in Urban Environment 4
       1.1.2 User Interface 5
       1.1.3 Adjustable Autonomy 6
   1.2 Motivation 6
   1.3 Challenges 7
   1.4 Contributions of My Work 7
   1.5 Guide 8

2 Background 9
   2.1 Related Literature 9
       2.1.1 Current Research in Cooperative Control 9
       2.1.2 Cooperation between Human and Automated Agents 11
       2.1.3 User Interface 13
   2.2 Previous Work Done at MAGICC Lab 13
   2.3 Summary 14

3 Path Generation and Data Processing 15
   3.1 Path Generation 15
5.2.2 Simulation Results .............................................. 59
5.3 Summary ......................................................... 63

6 Conclusions and Future Work ....................................... 65

Bibliography ............................................................ 74
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Levels of automation of decision and action selection</td>
<td>11</td>
</tr>
<tr>
<td>4.1</td>
<td>UI design criteria</td>
<td>32</td>
</tr>
<tr>
<td>4.2</td>
<td>Design of experiments, an example of a running order</td>
<td>40</td>
</tr>
<tr>
<td>4.3</td>
<td>Repeated measures ANOVA: Response time</td>
<td>41</td>
</tr>
<tr>
<td>4.4</td>
<td>Repeated measures ANOVA: Adjustment</td>
<td>44</td>
</tr>
<tr>
<td>4.5</td>
<td>Repeated measures ANOVA: Quality of the operational team cost</td>
<td>46</td>
</tr>
<tr>
<td>4.6</td>
<td>Repeated measures ANOVA: Quality of the operational team cost. The data is obtained using without spatial interface.</td>
<td>47</td>
</tr>
<tr>
<td>5.1</td>
<td>Simulation parameters used in simulations</td>
<td>58</td>
</tr>
</tbody>
</table>
List of Figures

1.1 Examples of unmanned aerial and ground vehicles ........................................ 2
1.2 Example of urban environment. ................................................................. 5
3.1 Path generation process diagram. ............................................................... 16
3.2 Example of the Voronoi diagram. ............................................................... 17
3.3 Before and after pruning of the Voronoi diagram. ................................. 18
3.4 Example of the hazardous zone. ............................................................... 19
3.5 The locus of points of the coordination function. ....................................... 23
3.6 The sorted locus of points for the vehicle $i$. ........................................ 24
3.7 Example of simultaneous arrival. ............................................................. 25
3.8 Example of tight sequence arrival. ............................................................ 27
3.9 Example of loose sequence arrival. ........................................................... 28
4.1 Example of GUI used in interface testing. .................................................. 33
4.2 Comparison of the temporal control and the coordination variable/function control. ................................................................. 34
4.3 Example of spatial control interface used in testing. ............................... 36
4.4 Example of cost control interface. ............................................................. 36
4.5 Example of NASA TLX used to measure the workload of each interface. ... 38
4.6 Two levels of map settings used in testing. .............................................. 38
4.7 Example of the constrained path case. ...................................................... 39
4.8 Mean response time of interfaces. ............................................................ 42
4.9 Relationship between mean response time, interface type, and constraint type. ................................................................. 43
4.10 Results of workload measured. .............................................................. 44
4.11 Mean operational team cost of interfaces. .............................................. 46
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Environment of graphical user interface</td>
<td>50</td>
</tr>
<tr>
<td>5.2</td>
<td>Coordination variable/function interface</td>
<td>50</td>
</tr>
<tr>
<td>5.3</td>
<td>Examples of changes in variables based on the ETA.</td>
<td>52</td>
</tr>
<tr>
<td>5.4</td>
<td>Example of the spatial map display.</td>
<td>53</td>
</tr>
<tr>
<td>5.5</td>
<td>Vehicle cost analysis windows.</td>
<td>54</td>
</tr>
<tr>
<td>5.6</td>
<td>Variable option selectors.</td>
<td>55</td>
</tr>
<tr>
<td>5.7</td>
<td>Example of the cooperative timing mission simulation.</td>
<td>60</td>
</tr>
<tr>
<td>5.8</td>
<td>Result of simultaneous arrival simulation.</td>
<td>61</td>
</tr>
<tr>
<td>5.9</td>
<td>Results of sequential arrival simulations.</td>
<td>62</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Applications for unmanned vehicles have been emphasized recently. The main advantage of using unmanned vehicles over conventional manned vehicles is that no human life is put in danger during missions in hostile environments because human operators can control these vehicles from remote locations. Hence, no human lives are directly threatened in the event that a vehicle is attacked or destroyed. Figure 1.1 illustrates several unmanned aerial and ground vehicles used currently. It is now common to see unmanned ground vehicles (UGVs) used during bomb threats or volcanic activity, while unmanned aerial vehicles (UAVs) successfully serve many functions in the battlefield. Consequently, unmanned vehicle applications have been proposed for use in many applications such as environmental monitoring, reconnaissance, and military missions [1, 2].

Recently much of the research interest in unmanned vehicles has shifted from controlling a single vehicle to managing multiple vehicles. For instance, when we want to coordinate rendezvous for multiple vehicles, many factors must be considered. Each individual vehicle will have different initial conditions and be in different situations (e.g., vehicle damage, remaining fuel amount, and unknown threat condition). In order to operate multiple vehicles, many issues (e.g., vehicle control and mission control) must be considered simultaneously. Among these considerations, the coordination among these vehicles is of interest because it is essential to successfully controlling multiple vehicles that share a common team goal.

Coordination among these vehicles becomes more difficult in a dynamically varying environment. In general, as more vehicles are added to the existing system, the system is required to handle more information. To effectively utilize available information, it is
Figure 1.1: Examples of unmanned aerial and ground vehicles [3, 4, 5, 6].
important to pass the right information to the right places, including information sent by and passed to human operators. Efficient and judicious information processing is one of the keys to a successful mission.

Generally, computers process data more quickly than humans for known situations. In many autonomous systems, algorithms have been implemented based on carefully designed and tested mathematical models. Nonetheless, computers still have cognitive and situational awareness limitations. The following question arises: Can human input improve computer performance when dealing with situation awareness and cognitive problems? To answer this question, human understanding, knowledge, and experience into the decision making process of an existing autonomous multi-vehicle system will be integrated to improve and enhance overall performance.

1.1 Problem Statement

The cooperative timing problem allocates multiple UGVs cooperatively to accomplish timing-sensitive missions. In cooperative timing problems, it is important to ensure that certain timing constraints are met during execution. Each vehicle has to maneuver in space to arrive at its target while avoiding threats and satisfying timing constraints. To do so, precise calculations and preparations are required. During this phase, humans hypothesize and prepare for any situation (e.g., pop-up threats) they might encounter as they strive to achieve the primary goals of the mission.

Humans have utilized computers to solve these kinds of complicated problems and, in many cases, computers have produced superior results compared to humans in jobs ranging from gathering information to executing a set of instructions. But, are programmed algorithms enough to compensate for human experience? Can we integrate human experience into the system to improve the overall outcome? To answer these questions, it is hypothesized that combining human involvement with the existing computer system can lead to more efficient solutions. In order to prove this hypothesis, the cooperative timing of multiple UGVs will be used as its application. In this task, a user interface will be created to integrate human inputs with existing computer algorithms. This will allow the user to
participate in information analysis and decision making processes along with two levels of adjustable autonomous systems.

1.1.1 Cooperative Timing Problem in Urban Environment

The main objective of a cooperative timing problem is to ensure that multiple vehicles arrive at their designated targets or destinations at times corresponding to a prearranged schedule. The problem is more difficult when vehicles have to maneuver in highly populated urban environments instead of open spaces. One of the main differences between open space and urban environment scenarios is that all vehicles have to avoid inaccessible areas (i.e., buildings, and road blocks) known as “no go zones” in an urban environment scenario. The field in the urban environment scenario consists of three areas: a safe zone, a hazardous zone, and a no go zone. The no go zone represents obstacles that are not accessible by any of vehicles. The hazardous zone is where the environmental threat is higher than other places in the field. In this urban scenario, this hazardous zone could represent direct exposure to an enemy. The rest of the field, excluding the no go zone and the hazardous zone, is the safe zone.

Each possible path is divided into two sections: safe and dangerous. If a path contains a path segment that goes through the hazardous zone, this path is considered to have a dangerous path segment. The total safe path section length can be calculated from the starting point to the last waypoint before entering the hazardous zone. If the path does not contain any dangerous path section, the total path length is equal to the safe path section length. The total dangerous path segment is computed from the first segment that enters the hazardous zone to the last waypoint. One of the main constraints for the problem is that if the chosen path contains a dangerous path section, the vehicle should move with its maximum velocity to avoid the total time of exposure to the direct enemy threat. By adjusting the vehicle velocity during the safe path section, it is possible to meet cooperative timing constraints.

Figure 1.2 illustrates an example of an urban environment scenario. Note that each vehicle’s path may contain a hazardous area. In addition, once a vehicle is exposed to an arbitrarily located threat, the rest of their path is in jeopardy. If exposed paths are
unavoidable, the vehicle at risk should move as quickly as possible to minimize time near a threat while the rest of the vehicles must compensate by varying their speed or path length to meet the timing constraints.

In an urban environment cooperative timing scenario of this type, each vehicle typically has one destination, but multiple vehicles can be assigned to the same target. In addition, these vehicles have two major constraints. First, all vehicles must meet timing constraints. Second, all vehicles must plan to avoid unapproachable areas while maneuvering. Thus, all vehicles must be controlled individually and coordinated as a team to accomplish their objectives, while also satisfying their timing and accessibility constraints. This research will investigate three different timing constraints for cooperative timing problems in urban environment scenarios while properly managing vehicle paths to avoid any maneuvering challenges.

1.1.2 User Interface

One of the keys to successfully integrating human experience is to have an adequate user interface (UI). Decades ago, most computers were built under the assumption that all users fully understood the system. However, many computer systems have recently become much more sophisticated and complicated. Currently, computers are not only for specialists, but also for general users. The UI supports all users from these systems and
increases their overall performance to make achieving their goals easier. Hence, having an efficient UI is important for improved communication between computers and users. To integrate humans into an adjustable autonomy system, an effective UI is required. The UI can take the form of text, graphics, or a mixture of text and graphics. In my research, an efficient UI will be thoroughly developed to organize and present complex information to humans in the cooperative timing problem discussed earlier.

1.1.3 Adjustable Autonomy

In this research, two primary items are of interest. First, multiple levels of autonomy, which assist human operators to plan a cooperative timing problem, have been developed. Then, a user interface that helps users to utilize the multiple levels of autonomy has been developed. To attack the problem, the concept of adjustable autonomy (AA) has been adapted. By adapting this concept, human inputs to the cooperative control system are integrated successfully. In this research, two levels of autonomy (Fully autonomous mode and Semi autonomous mode) have been applied.

1.2 Motivation

During cooperative control experiments, users must be prepared to react quickly to any possible scenario. Unfortunately, it is nearly impossible to enumerate and prepare for all possible situations. Allowing human knowledge to be input into a system will provide a way to broaden the types of situations that can be dealt with effectively. One method for sharing information between the computer and human operator that will be explored is that of coordination variables and coordination functions [7].

The coordination variables and functions were developed to handle complex information in cooperative control applications. Reference [7] provides the fundamental ideas of these coordination variables and functions. Coordination variables are defined as a single vector quantity of information that “must be jointly shared to facilitate cooperation.” This information is intended to be minimal, yet contain all of the information needed for the coordination. Coordination functions are used “to parameterize the effect of the coordination variable changes on the myopic objectives of each agent.” These coordination functions
and variables are currently used in cooperative control applications such as cooperative timing problems.

This approach will allow a system to provide adequate information to users in a short time period to aid them in making decisions which lead to an improvement in overall performance. In this way, performance objectives can be met, while satisfying given constraints.

1.3 Challenges

One of the important challenges to overcome is the complexity in the system. The system that is going to be used in this research has very sophisticated structural layers. This system integrates many subsystems such as waypoint path planning, trajectory generation, trajectory tracking, and low-level robot control.

In addition, timing is the most crucial constraint. Managing synchronization and interaction among multiple vehicles is another important task to handle. Since one vehicle’s behavior affects the rest of the team, there is no guarantee that timing constraints will be satisfied unless proper action is taken. Consequently, a solution chosen by a user must be carefully reviewed and tested to coordinate with the other elements of a complete solution.

Furthermore, vehicle dynamic limitations must be considered during planning and coordination. In other words, vehicle dynamics must be considered to avoid infeasibility among solutions. For instance, any turning path generated by the trajectory generator must be feasible for the vehicle to execute. Hence, this kind of problem must be addressed in the development of a cooperative control system. One of the biggest challenges is developing an efficient UI. All users should be able to interact with the control system easily so that it guides them to make the best decisions.

1.4 Contribution of My Work

This thesis makes contributions to several aspects of timing-sensitive cooperative control applications including

- The development and testing of numerous user interface options,
• The development of path generation and coordination data processing, and

• The development and implementation of multiple levels of adjustable autonomy.

First and foremost, the graphical user interface (GUI) proposed in this thesis enables users to make decisions based on temporal, spatial, cost, and coordination variable/function control interfaces to allocate a team of unmanned ground vehicles in an urban environment. The efficiency of each of the four interface options was determined quantitatively by measuring different metrics such as response time, workload, and quality of decisions.

In addition, the work in this thesis contributes to path generation and coordination data processing. This research examines the feasibility of utilizing the Voronoi graph search and Eppstein’s algorithm for path planning in the urban environment. Furthermore, this thesis successfully demonstrates the implementation of coordination variables and functions in a novel way to represent critical coordination data to users controlling a multi-vehicle system through a GUI.

This research also makes contributions in the area of adjustable autonomy. Drawing an adjustable autonomy descriptions in the literature, this thesis implements multiple levels of autonomy in the form of a GUI that helps users plan cooperative timing problems.

1.5 Guide

Chapter 2 of this thesis explores the background information related to this work. Path generation algorithms and data processing methods are presented in Chapter 3. Newly developed user interfaces and their validation are described in Chapter 4. Chapter 5 consists of the features of the proposed interface and results of cooperative timing problems in urban environments. Chapter 6 concludes with results and presents recommendations for future work.
Chapter 2

Background

2.1 Related Literature

There are many research topics that have been explored concerning cooperative control applications and their implementations. In this section, I will focus on four major areas. First, I will review the current research on cooperative control. Second, I will discuss cooperation between human and automated agents. Third, I will discuss user interface in general. Fourth and last, I will discuss the current work at Brigham Young University MAGICC\textsuperscript{1} Laboratory.

2.1.1 Current Research in Cooperative Control

Within the field of cooperative control applications, research on formation control and path planning for multiple vehicles has recently increased. In [8], an architecture for formation control of multiple spacecraft is investigated. This article claims that a multiple spacecraft approach may reduce costs and strengthen overall robustness as compared to a single large spacecraft. This article explores three different formation control architectures, leader-follower, behavioral, and virtual-structure and explains the benefit of each. Egerstedt et al. [9] discuss the feasibility of control over formation architecture. They first validate their control methods with an individual robot and then expand the same concept to the different formation structures referred to as translational rigid body motions.

Coordination structures are discussed in [10]. To achieve the coordination objective, this article suggests three items must be considered: path planning, trajectory generation,

\textsuperscript{1}Multiple AGent Intelligent Coordination and Control
and trajectory tracking. This article introduces an approach using Voronoi diagrams to determine all feasible solution sets for the path planning. Then the trajectory generator applies UAV dynamics to generate trajectories flyable by UAVs. Finally, the UAVs controlled by the trajectory tracker actually fly the trajectories.

Along with these three items, McLain et al. [11, 12, 13] add another constraint, timing. One emphasis of their work is in timing-critical missions. These articles focus on the estimated-time-of-arrival (ETA) of UAVs at designated targets. Examples of timing-critical missions include rendezvous or simultaneous arrival, and sequential arrival. Interestingly, these timing-critical missions have been identified by the Air Force as an area for future research emphasis. In [1], Brigadier General Daniel P. Leaf emphasizes the importance of cooperative UAV flight, specifically in timing critical missions, based on his experience as a commander. He identifies this capability as “rolexing—the ability to adjust mission timing on the move to compensate for inevitable changes to plans and still make the time-on-target” [1, pg. 53].

Cooperative path planning for multiple vehicles is another example of cooperative planning that has been explored [10, 14, 15]. The common theme of these articles is generating cooperative paths for multiple UAVs in dynamic and uncertain environments. Bellingham, et al. [14] approach this problem stochastically. First, they prepare for dynamically changing environments such as the addition of waypoints, the loss of a UAV, or changing obstacle positions. Then they analyze the uncertain environment and optimize it using stochastic methods. In [15], an agile autonomous vehicle environment has been investigated and feasible vehicle paths have been presented. The main idea of this article is to construct nodes and milestones, then carefully evaluate the cost and feasibility at each milestone. After the computation, the trajectory from the initial condition to the milestone is determined and the process is repeated. Interestingly, this article suggests a new motion-planning algorithm, known as a Probabilistic RoadMap (PRM) planner, which generates a graph of feasible paths and connects different locations in the workspace.

Another focus in cooperative control applications is cooperative planning for multiple agents. Recent research [15, 16, 17] indicates the importance of cooperative planning. Earlier research [18, 16, 19] in this area was targeted on problem-solving techniques in
The work of Layton, et al. [19] focused on improving automated solutions referred to as cooperative, which can be brittle in unexpected situations. Lauzac, et al. [16] concentrate their work on how to react in a dynamically changing environment. They also investigate ways to assist individual agents that have awareness and cognitive limits.

### 2.1.2 Cooperation between Human and Automated Agents

Automation impacts our lives in areas ranging from home automation [20] to air traffic control [21]. New automation applications such as advanced automation for motor vehicles [22], decision support in medical systems [23], and cockpit automation [24] are currently being explored. Automation has become more common in our lives due to explosive growth in both the hardware and software fields [25]. Accordingly, Sheridan, et al. in [25] define the levels of automation in decision and action selection. As illustrated in Table 2.1 [25, pg. 287], these 10 levels describe a varying level of human involvement in the automation process.

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>The computer decides everything, acts autonomously, ignoring the human.</td>
</tr>
<tr>
<td>9</td>
<td>Informs the human only if it, the computer, decides to</td>
</tr>
<tr>
<td>8</td>
<td>Informs the human only if asked, or</td>
</tr>
<tr>
<td>7</td>
<td>Executes automatically, then necessarily informs the human, and</td>
</tr>
<tr>
<td>6</td>
<td>Allows the human a restricted time to veto before automatic execution, or</td>
</tr>
<tr>
<td>5</td>
<td>Executes that suggestion if the human approves, or</td>
</tr>
<tr>
<td>4</td>
<td>Suggests one alternative</td>
</tr>
<tr>
<td>3</td>
<td>Narrows the selection down to a few, or</td>
</tr>
<tr>
<td>2</td>
<td>The computer offers a complete set of decision/action alternatives, or</td>
</tr>
<tr>
<td>1</td>
<td>The computer offers no assistance: human must make all decisions and actions.</td>
</tr>
</tbody>
</table>
Parasuraman and Riley [26] researched automation from the aspect of human performance. They claimed that one important reason not to remove human operators from automated systems was the common perception that humans are more flexible, adaptable and creative than computers. Hence, humans can anticipate and react better to unexpected situations.

The discussion table above leads to a new term: *adjustable autonomy* (AA). AA has been dealt with and discussed in many articles [27, 28]. In [29], Dorais, et al., define AA. The main difference between AA and conventional autonomy (such as executing predefined fixed algorithms), is that human operators are considered as being in the environment; they are communicating with and involved in the system. In [30], a prototype of AA was established by constructing multiple levels of autonomy. Along with this prototype, many architectures, such as a communication protocol, robot server, and control program, were built and tested to show the effectiveness of AA. Falcone, et al. [31] define points of adjustability between humans and robots. According to this article, the adjustability is *bilateral* because both the user and the autonomous agent have the ability to adjust to environmental change. It is also bidirectional and multidimensional due to the various levels of autonomy available.

Scerri, et al. [32] address the issue of deploying UAVs in the real world from the perspective of AA. This article focuses on AA coordination challenges, especially during the decision making process. Related to AA, *mixed*-initiative is another method of human-system interaction. Allen, et al. define mixed-initiative as a flexible interaction strategy [33]. In [34], the possible challenges of automation and the strengths of mixed-initiative interaction are discussed. Designers of automated systems struggle with setting the levels of autonomy. This article addresses the possible human challenges in maintaining awareness in dynamic, multi-tasking control environments due to an increase of complexity and a decrease of human initiative. *Mixed*-initiative control, however, frees humans from excessive workloads and allows them to focus on more valuable activities including high-level strategizing, visual information processing, and analytical reasoning.
Kortenkamp, et al. [35] investigate mixed initiatives by constructing a scale of mixed initiative control levels. This article describes a software architecture for autonomous control consisting of five levels: teaming, supervisory, traded, guided, and teleoperation. Using this approach, users can switch between supervisory and guided modes to achieve the desired level of mixed initiative control. This mode was described as the traded mode. In traded control, humans and computers can easily switch between control modes. Under typical conditions, the robots perform autonomously, but humans have the capability to override the system to complete subroutines or rescue robots from dangerous situations or failures.

2.1.3 User Interface

Goodrich, et al.[30] assert that having an efficient interface is important for human involvement in computer systems. Adams and Paul [36, 37] introduce four levels of human supervisory control: task, regulation, processing and data levels. For each level, they describe the human involvement in the system. Within each of these levels, they enable human interaction with the system by creating appropriate user interfaces.

Cheng, et al. [38] also indicate the importance of the supervised autonomy framework. By generating a simple yet effective interface, they successfully demonstrate a framework for supervised autonomy. In [39, 40, 41, 42] user interfaces have been emphasized. These articles introduce a variety of interfaces, from a dialogue-based interface [42] to personal digital assistant (PDA) drivers [40]. In addition to describing the specifics of the user interface, these articles also describe user interface architecture and control modes. Ciancarini et al. [39] also demonstrate another user interface application using the World Wide Web (WWW). In this article, the relationship between a client and a server is described as well as its implementations on the WWW.

2.2 Previous Work Done at MAGICC Lab

The main research activities of the Multiple AGent Intelligent Coordination and Control (MAGICC) Lab are mobile robots control and cooperative control of multiple-vehicle systems. During the last several years the MAGICC Lab has developed algorithms
and architectures, allowing mobile robots and UAVs to maneuver autonomously. Previously Kelsey developed the MAGICC Mobile Robot Toolbox (MMRT) [43] that allows users to control mobile robots under Matlab Simulink. MMRT provides a convenient interface for rapid implementation and testing of mobile robot control applications. In addition, a group of students developed a “Capture the flag” game with multiple omni directional robots to demonstrate the coordination in a multiple robot system. In this game, a simulated UAV is flying over the designated area and sending intelligence to the ground robots. The ground robots then utilize this information as well as their own sonar information, and try to capture the enemy flag while avoiding enemy robots. Most recently MAGICC Lab work has focused on controlling single and multiple UAVs from a remote location using a virtual cockpit program and has successfully demonstrated cooperative control in many different scenarios.

2.3 Summary

This chapter presented current research in four areas: cooperative control, adjustable autonomy, user interface, and previous work done at MAGICC Lab. In each section, various advanced theories and techniques are realized and a new direction for my research has been determined. In the next two chapters this information will be applied to complete an urban environment cooperative timing problem.
Chapter 3

Path Generation and Data Processing

This chapter will present a number of methodologies used in this research. In the first section, a method of a path generation will be presented. The second section will discuss the coordination variables and functions and their application in an urban environment cooperative timing problem.

3.1 Path Generation

In order to control multiple vehicles following a certain path, path generation is an essential component that must be provided prior to making a final path selection. Before this can be done, the known environmental parameters (i.e. threat location, hazardous area, and safe nodes) must be provided. The Voronoi graph \[44\] method is first used to construct sets of safe path nodes. An Eppstein algorithm is then used to search the Voronoi graph to find the best path options\[45\]. Figure 3.1 indicates the general process diagram for the path generation. The following sections will discuss each process in detail.

3.1.1 Voronoi diagram

The approach used to construct nodes for the path planning was the Voronoi diagram. Given a set of points in a plane to be avoided, the Voronoi diagram is a graph that divides the plane into convex cells, each containing a point inside in the given space. Any plane that contains \(n\) points will result in \(n\) convex cells containing one point in each cell. Figure 3.2 shows an example of the Voronoi diagram. Note that each straight line segment is called an edge while the two end points of each edge are called nodes.
Although the main advantage of using a Voronoi diagram to generate edges and nodes is the computational speed [46], it requires a few post-processing steps including the pruning of infeasible edges. Since this research is focused on urban environments, all buildings are defined as sets of polygons containing multiple points to be avoided. Once all nodes and edges are identified by the Voronoi algorithm, the pruning algorithm is performed to eliminate all edges that cross any side of the polygons. Such edges are infeasible. Every edge is tested for intersecting with all sides of every obstacle. If they intersect, the corresponding edge is removed from the Voronoi diagram. Figure 3.3 shows the Voronoi diagram before and after pruning. Notice that all edges located inside of the polygon obstacles are not pruned in order to reduce the computational load. However, all edges connecting others outside of the obstacles have been pruned. As a result, those edges inside of the obstacles will not affect path generation.
3.1.2 Path Generation

Once all feasible nodes and edges are identified, it is necessary to determine the cost of each path segment. In this research, the Eppstein search algorithm is used to generate feasible paths for vehicles. Prior to using this algorithm, a cost matrix representing the connectivity of nodes and their associated cost must be generated and passed to the Eppstein search algorithm. The main purpose of generating a suitable cost matrix is to have the Eppstein algorithm return as many feasible paths as possible.

The cost matrix is a symmetric square matrix and has a size of the total number of nodes. As mentioned previously, each feasible edge contains two nodes. Based on this node information, each entry of the matrix is represents the connectivity of nodes. If a feasible edge has two nodes, \( i \) and \( j \), the cost of an edge is calculated and entered to the matrix entries \( i j \) and \( ji \). Otherwise, other entries are assigned as \(-1\) representing no connection between nodes.

The cost of each edge is determined based on two factors: a length of each edge and exposure to the hazardous zone. All feasible edges are tested to determine if they are
potentially dangerous paths. Figure 3.4 shows an example of hazardous zone construction. Note that the area highlighted by red indicates the hazardous area while blue rectangles represent the urban obstacles (i.e., buildings). If either of the two nodes of an edge is inside of the hazardous zone, it is considered to be dangerous. If an edge crosses any side of the hazardous zone, it is also considered as a dangerous. After identifying the danger of all edges, the costs of all feasible edges are calculated. The cost is determined by a combination of the length and the danger of an edge. All edges exposed to the hazardous zone are penalized and consequently the cost of connectivity increases.

Once the cost matrix is determined, it is passed to the Eppstein algorithm along with the start point, the end point, the cost matrix, and the number of desired paths, \( k \). The Eppstein’s algorithm then determines the \( k \) lowest cost paths between the start point and the end point by analyzing the cost matrix. After computing the \( k \) best paths, each path is searched node by node to determine if the path passes through the hazardous zone. If the path contains a segment crossing the hazardous zone, the path information and the information for the first node crossing the zone are stored. This information then is used to coordinate vehicles and satisfy timing constraints using coordination variables and functions.
Figure 3.4: Example of the hazardous zone. The dark gray objects are obstacles while the red polygon indicates the hazardous zone.

3.2 Coordination Variables and Functions

This section contains the details of coordination variables and functions introduced in [7], upon which this research builds. For the purpose of clarification, the following section draws heavily from [7], including notation and figures. The definition of the coordination variables and functions, as well as the technical approach to their application will be presented.

The main concept of coordination variables and functions is that the minimum amount of information to achieve a common cooperation objective should be identified and shared among vehicles in the team. Let $\chi_i$ be the situation state space for the $i^{th}$ vehicle on the team and $x_i \in \chi_i$ be the situation state of the $i^{th}$ vehicle. The situation state in the given cooperative timing problem includes a vehicle’s current velocity, position, timing constraints (i.e. time intervals between vehicle’s estimated time of arrival (ETA)), and environmental parameter (i.e. hazardous area). Note that each vehicle can act to influence the effectiveness of the team. For a given situation $x_i$, the set of feasible decisions for a vehicle is given by $U(x_i)$ and let $u_i \in U_i$ be the decision variable for the $i^{th}$ vehicle. The
chosen set of decision variables will determine the quality of the cooperation achieved. The
decision variables for cooperative timing in the urban scenario consist of path waypoints and velocity information.

Coordination variables contain the minimum amount of information needed by the
team for coordination. The underlying concept of coordination variables is that all vehicles in the team share the coordination variables and respond appropriately to achieve cooperation objectives. If $f_i : \chi_i \times U_i \rightarrow \mathbb{R}^c$ is a function that maps situation state and decision vector pairs to coordination space $\mathbb{R}^c$, then the set of feasible coordination variables for the $i^{th}$ vehicle in situation state $x_i$ is given by

$$\Theta_i(x_i) = \bigcup_{u_i \in U_i(x_i)} f_i(x_i, u_i)$$

Note that $\Theta_i(x_i)$ is not necessarily a connected set. In addition, the local cost of an individual vehicle in the team is represented as,

$$\phi_i(x_i, \theta) = J_i(x_i, f_i^+(x_i, \theta)), \quad (3.1)$$

where $f_i^+$ is a pseudo inverse of $f_i$.

The function, $J_i(x_i)$, given by Equation 3.1 is called the coordination function of the $i^{th}$ vehicle. In this research, the cooperative problem of interest is to minimize a team objective function that consists of all individual vehicle cost functions. If $J_T : \mathbb{R}^N \rightarrow \mathbb{R}$ is defined as the team objective function, then a general cooperative timing problem can be described as,

$$(u_1, \ldots, u_N) = \arg \min_{U_1 \times \cdots \times U_N} J_T(\phi_1(x_1), \theta), \ldots, \phi_N(x_N, \theta)), \quad (3.2)$$

subject to

$$f_i(x_i, u_i) = f_j(x_j, u_j) + \Delta_{ij} + \tau_j, \quad \forall i, j \in 1, \ldots, N,$$

where $\Delta_{ij}$ is the time offset between $i^{th}$ and $j^{th}$ vehicle, and $\tau_j$ specifies the duration of the desired arrival time window of $j^{th}$ vehicle. Notice that this optimization problem could have computational problems as the number of vehicles increases.

Using coordination variables and coordination functions, the optimization problem of Equation 3.2 and 3.3 can be represented as 3.4,

$$\theta^* = \arg \min_{\theta \in \cap_i \Theta_i(x_i)} J_T(\phi_1(x_1, \theta), \ldots, \phi_N(x_N, \theta)).$$

(3.4)
Finally, after a team optimal value using coordination variables and functions is found, the individual vehicle’s decisions (i.e. specific path and vehicle’s velocity) can be found by solving for the influence variable from the relationship

\[ u_i = f_i^\dagger(x_i, \theta^*). \]

### 3.2.1 Application to Cooperative Timing in Urban Environment

The situation state space \( \chi_i \) for cooperative timing in urban environment problems is composed of vehicle initial position vectors, a final destination position vector, and the set of path. Note that

\[
x_i = \begin{pmatrix} z_{i0} \\ z_{zf} \\ H_i \end{pmatrix},
\]

where \( z_{i0} \) is the initial position of the vehicle, \( z_{zf} \) is the destination position, and \( H_i \) is the environmental parameters including the building locations and the hazardous zone information.

In this problem set up, the vehicle velocity and the set of waypoints describing any possible path are the decision vectors \( U_i(x_i) \) at \( x_i \in \chi_i \). The vehicle velocity is described as

\[ v_{min} \leq v_i \leq v_{max} \]

where \( v_{min} \) is the minimum vehicle velocity, \( v_i \) is the actual vehicle velocity, and \( v_{max} \) is the maximum vehicle velocity. The feasible vehicle velocity is bounded by the minimum and the maximum velocity. Hence a decision vector can be presented as

\[
u_i = \begin{pmatrix} v_i \\ W_i \end{pmatrix},
\]

where \( v_i \) is a feasible vehicle velocity, and \( W_i \) is a set of feasible waypoints for path \( i \). A set of waypoints, \( W_i \) can be represented as

\[ W_i = \{w_{i1}, w_{i2}, \ldots, w_{ip}\}, \]
where \( w_{i1} = z_{i0} \), and \( w_{iP} = z_{if} \).

In this urban environment cooperative timing problem, the ETA of each vehicle is the coordination variable. The ETA is determined for a given path when a vehicle is following the waypoints, \( W_i \) at a velocity of \( v_i \). Notice that a vehicle velocity \( v_i \) consists of two different velocities: the safe path section velocity \( (v_{is}) \) and dangerous path section velocity \( (v_{id}) \). \( v_{is} \in [v_{min}, v_{max}] \) is the main variable that determines the ETA of the \( i^{th} \) vehicle, while \( v_{id} \) is set to be a maximum velocity, \( v_{max} \), to minimize the total time of exposure in the hazardous area.

For any path \( W_i = \{w_1, w_2, \cdots, w_p\} \), the length of the path is given by

\[
L(W) = L_s(W_s) + L_d(W_d) = \sum_{j=2}^{k-1} \|w_j - w_{j-1}\| + \sum_{j=k}^{p} \|w_j - w_{j-1}\|, 
\]

where \( L(W) \) is the length of the given path, \( L_s(W_s) \) and \( L_d(W_d) \) are the length of the safe and dangerous paths respectively, and \( k \) is the index number of the given path where the dangerous nodes start.

For all vehicles, the value of the coordination variable is formulated from the state and decision vectors,

\[
\theta_i = f_i(x_i, u_i) = L_{is}(W_{is})/v_{is} + L_{id}(W_{id})/v_{max}. \tag{3.5}
\]

The local objective function, \( J_i \), is given by a linear combination of total time of exposure in safe and dangerous path segments and the energy consumption determined by vehicle velocity in a given path:

\[
J_i(x_i, u_i) = K_s \left( \frac{L_{is}}{v_{is}} \right) + K_d \left( \frac{L_{id}}{v_{max}} \right) + K_e \left( L_{is} \cdot v_{is} + L_{id} \cdot v_{max} \right), \tag{3.6}
\]

where \( K_d > K_s > 0 \) and \( K_e > 0 \) are constants. This assumes that vehicle energy consumption is proportional to the ground resistance determined by a vehicle’s speed.

Once the local objective function, \( J_i \), is constructed, the pseudo inverse, \( f^\dagger(x_i, \theta) \), must be calculated in order to determine the proper trajectory \( u_i \in U_i(x_i) \). Choosing the correct \( u_i \) is essential to meet a specified time of arrival, \( \theta \in \Theta_i(x_i) \) for a given situation state \( x_i \). If \( U \) and \( \Theta \) have a one-to-one relationship, the coordination function \( \phi_i \) is equal to the local objective function, \( J_i \). However, in most cases, the desired time of arrival can be
achieved by more than one trajectory. In other words, there is more than one \( u_i \in U_i \) that satisfies \( \theta = f(x_i, u_i) \), which has a local objective function of \( J_i(x_i, u_i) \). Figure 3.5 illustrates the locus of points, \( \bigcup_{u_i \in U_i} J_i(x_i, u_i) \). Note that the range of ETA, \( \theta \), on each trajectory is different based on the total length of the trajectory and the presence of the dangerous path section on the trajectory.

![Diagram](image)

**Figure 3.5:** The locus of points (\( \theta \) versus \( J_i(x_i, u_i) \)).

The total ETA range of the \( i^{th} \) vehicle can be determined as the union of all \( \theta \) of the \( i^{th} \) vehicle:

\[
\Theta_{range} = \bigcup_{n=1}^{P} f_i(x_n, u_n),
\]

where \( P \) is the total number of trajectories of vehicle \( i \).
The next step is to determine the minimum cost trajectory, $u_i \in U_i$, at a given ETA. Since the primary objective of this method is to determine the minimum cost for the team of vehicles, the minimum cost trajectory can be found by taking a pseudo-inverse of $\theta = f_i(x_i, u_i)$. The pseudo-inverse, $f_i^\dagger(x_i, \theta)$ of $f_i$, is described as the following:

$$f_i^\dagger(x_i, \theta) = \arg \min_{u_i \in U(x_i)} J_i(x_i, u_i)$$

subject to

$$\theta = f_i(x_i, u_i).$$

Once $f_i^\dagger(x_i, \theta)$ is found, it can be entered to Equation 3.1 to determine all paths satisfying the minimum cost trajectory. Figure 3.6 illustrates the result of sorting all paths for the vehicle $i$.

![Figure 3.6: The sorted locus of points for the vehicle $i$.](image)

### 3.2.2 Simultaneous Arrival Constraints

When a simultaneous arrival timing constraint is chosen, all vehicles must arrive at their destinations at the same time. The timing constraint for simultaneous arrival can be stated as
where $T_s$ is the arrival time for all vehicles. As shown in Figure 3.7, ETAs for a team of three vehicles are same. In other words, all timing constraint parameters defined in Equation 3.3 (e.g. $T_{12}$, $T_{23}$, $\tau_1$, $\tau_2$, and $\tau_3$) are zero. The objective of this timing problem is to select a solution that minimizes the sum of individual costs while satisfying the simultaneous arrival timing constraint. Since the coordination function of each vehicle $J_i(x, u)$ in this problem has a parabolic characteristic, the minimum value will typically lie at the vertex in the center of the coordination variable range. There is no guarantee that all vertices for all functions will be located at one place. Therefore, the sum of all vehicle costs, $J = \sum_{i=1}^{n} J_i(x, u)$, must be computed within coordination variable range to determine the minimum team cost.

![Figure 3.7: Example of simultaneous arrival.](image URL)
3.2.3 Sequential Arrival Constraints

Sequential arrival constraints can be divided into two different cases: tight sequential arrival and loose sequential arrival. When the sequential arrival option is selected, all vehicles arrive at their destinations sequentially based on the predetermined order set by either the algorithm or the user.

**Tight Sequence Constraints**

If the tight sequence arrival constraint is selected, each vehicle’s estimated-time-of-arrival (ETA) is determined with user defined time offsets. The tight sequential arrival timing constraint can be stated as

\[ T_1 = T_s \]
\[ T_i = T_s + \Delta_i, \quad i = 2, \ldots, N, \]

where \( T_s \) is the arrival time for the first vehicle, and \( \Delta_i \) indicates the time offset between the first and \( i^{th} \) vehicle. In this case, Equation 3.5 must be modified as,

\[
\theta_i = f_i(x_i, u_i) = \frac{L_{is}(W_{is})}{v_{is}} + \frac{L_{id}(W_{id})}{v_{max}} + \Delta_i.
\] (3.9)

Similar to the simultaneous arrival case, the objective of this case is to find the minimum team cost. Once all \( \Theta_i(x_i) \) are calculated, \( \theta_{range} \) can be determined as follows:

\[
\theta_{range} = \bigcap_{n=1}^{P} \Theta_i(X_i).
\]

Figure 3.8 illustrates an example of a tight sequential arrival for a team of three vehicles. Note that \( 3! \) permutations exist to determine an order of arrival. However, all sequences may not available for the sequential arrival cases due to timing constraints. In Figure 3.8, a new notation is introduced:

\[ T_{ij} = \Delta_j - \Delta_i. \]

This \( T_{ij} \) notation is used to help users determine time offsets between vehicles. Note that \( T_{12} = \Delta_2 \) due to \( \Delta_1 = 0 \).
Loose Sequence Constraints

The loose sequential arrival adds more complexity to the tight sequential arrival option. This option provides for a desired window of arrival times to the sequential arrival problem. Figure 3.9 displays an example of the loose sequence problem with a team of three vehicles. Notice that $T_{ij}$ notation is also used in this case like the previous example.
Figure 3.9: Example of loose sequence arrival.

Timing constraints for loose sequential arrival can be described as

\[ T_s \leq T_1 \leq T_s + \tau_1 \]
\[ T_s + \Delta_i \leq T_i \leq T_s + \Delta_i + \tau_i, \quad i = 2, \ldots, N, \]

where \( T_s \) is the chosen ETA of the first vehicle, \( \Delta_i \) represents the time interval between the first and \( i^{th} \) vehicles, and \( \tau_i \) specifies the duration of the desired arrival time window. Note that \( T_s + \Delta_i \) indicates the low end of desired arrival time window while \( T_s + \Delta_i + \tau_i \) represents the high end of the window. Similar to the tight sequential arrival case, Equation 3.5 can be modified as follows

\[ \theta_i = f_i(x_i, u_i) = L_{is}(W_{is})/v_{is} + L_{id}(W_{id})/v_{max} + \Delta_i + \sigma_i, \quad (3.10) \]
where $\sigma_i \in [0, \tau_i]$ is a slack variable. In this case, arrival time for each vehicle $T_j$ can be expressed in the following form

$$T_s + \Delta_i \leq T_j \leq T_s + \Delta_i + \tau_i \quad i = 1, \ldots, N,$$

(3.11)

where $\Delta_1 = 0$. All other $\Delta_i$ and $\tau_i$ must be specified before computation.

### 3.3 Summary of Path Generation and Data Processing

This chapter has introduced the path generation method as well as the data representation method based on coordination variables and functions. Eppstein’s algorithm is used to search sets of safe path nodes constructed by the Voronoi graph method for the best path options. Coordination variables and functions are then used to plan trajectories for a team of unmanned ground vehicles. The following chapter discusses the development of the user interfaces utilizing some of methods discussed in this chapter. Designing and testing of the user interfaces are also discussed in the next chapter.
Chapter 4

Development of a User Interface

The previous chapter discussed a series of methods to allocate multiple ground vehicles (UGVs) cooperating to accomplish timing-sensitive missions in an urban environment. In this chapter, different methods for presenting cooperative timing scenarios to users will be considered. The main purpose of the user interface (UI) design is to assist the user in making better decisions to control multiple vehicles more effectively in a cooperative timing application. The UI is one of the key parts of this research. This chapter will discuss the types of UIs that have been developed and tested. The first section will discuss the UI design processes. The second section of this chapter will explain how a graphical user interface (GUI) has been developed. The last section of this chapter will discuss the quantitative results obtained from tests to determine the best methods for presenting cooperative timing information to users. Four different interface strategies were tested: temporal, spatial, cost, and coordination variable/function.

4.1 User Interface Design

User Interfaces are not stand-alone applications. Rather they are essential elements of most software packages that allow users to interact with complex programs. Nonetheless, it is not enough for UIs to be visually appealing; they must also be user friendly. UIs need to be consistent and fully functional [47]. Based on three fundamental principles of UI design [48], the design constraints of the proposed UI to the UGV coordination problem were carefully identified and determined. Table 4.1 indicates all design constraints. Among the many UI options, graphical user interfaces offer a graphical method of solving many problems. In addition, GUIs provide the functionality to vary problem parameters
and present the results graphically. For these reasons, GUIs have become popular and are utilized by the majority of computer users.

Table 4.1: UI design criteria

<table>
<thead>
<tr>
<th>Users must be able to</th>
</tr>
</thead>
<tbody>
<tr>
<td>- identify relationship between ETA and each vehicle’s cost</td>
</tr>
<tr>
<td>- manipulate ETA</td>
</tr>
<tr>
<td>- choose any ETA or path available</td>
</tr>
<tr>
<td>- make a visual confirmation of any decision made before a final decision</td>
</tr>
</tbody>
</table>

4.2 Developing GUIs Using MATLAB

The main objective behind developing the proposed GUI is to obtain easy, yet intuitive control of a group of mobile robots that may have different properties. In addition, these robots are restricted by timing constraints such as simultaneous and sequential arrivals. The GUI is to be designed to assist users to determine final paths and velocities of each mobile robot in variable scenarios by providing calculated cost data for the vehicles in a graphical display.

Previous cooperative timing problems (i.e., the multi-agent rendezvous problem [11]) were developed and solved using MATLAB and it was necessary to maintain interoperability with this software. MATLAB allows users to develop user friendly GUIs that are implemented using the UI controls [49, 50, 51, 52]. Because of these advantages, MATLAB was chosen as a GUI design tool for this research.

The MATLAB software package provides easy-to-use editing, debugging, and powerful graphic functions. In addition, MATLAB provides its own basic UI, including a GUI-building interface called GUIDE, which allows users to select and place custom UI features (i.e., graph, buttons, and text) into the main GUI frame and connect them to the GUI control programs, known as script files or m-files. All MATLAB graphics are object oriented
and each of these objects has its own identifier or handle. By placing and modifying these handles on a GUI, the proposed GUI was created and revised.

### 4.3 Comparison of Four Interfaces

In cooperative timing applications, users must make a decision to allocate multiple vehicles in a timely manner. In order to present information for cooperative timing problems to users, four different control interfaces were considered: temporal, spatial, cost, and coordination variable/function. These four types of interfaces are designed to assist users in making efficient decisions for the cooperative timing missions.

**Figure 4.1** illustrates the GUI used during the interface testing. Though all four interface controls are combined in the GUI, each interface is operated independently of the
others. The plotting area at the top-left corner of the GUI can be used as both temporal control and coordination variable/function control interfaces. The testing GUI is designed to display only the appropriate information according to the testing options. Figure 4.2 shows two usages of the upper left window. Though both interfaces have a vertical line indicating ETA, the coordination variable/function control interface displays more information to the window while the temporal control interface only displays the feasible ETA range.

![Comparison of two types of interfaces.](image)

(a) Example of the temporal control. (b) Example of coordination variable/function control.

### 4.3.1 Temporal Interface

The temporal control shown in Figure 4.2(a) only allows users to have control over the temporal domain, estimated time of arrival (ETA). This option enables users to make their decisions quickly by adjusting the ETA on the temporal control interface though users can also access spatial and cost information. In this case, both spatial and cost information become dependent on the temporal control decisions of the user. Users can choose an ETA by dragging and dropping the vertical bar anywhere within the ETA range indicated by vertical lines and arrows.
4.3.2 Spatial Interface

Spatial control allows a user to control only spatial information. By changing the spatial variable (paths of vehicles), users can choose a solution for the cooperative timing application. In this case, temporal and cost information depend on the chosen spatial configuration. As shown in Figure 4.3, the spatial information consists of the environmental objects (obstacles displayed in dark gray and hazardous zone displayed in red) as well as vehicle paths indicated with red, green, and blue lines. Similar to the previous interface, the cost plots and the temporal plot will change as the user changes the spatial information. In order to change any path of the three vehicles, a user can click one of three buttons for each vehicle located on the right side of the map.

4.3.3 Cost Interface

The cost control shown in Figure 4.4 permits users to have direct control over the cost of the operations. The cost is defined as the operational risk. The cost of each vehicle is determined by a combination of factors, such as the lengths of the safe and a dangerous portions of the path and vehicle velocity. By manipulating the operational cost of each vehicle directly, users can make a decision for the cooperative timing application. The cost control interface consists of two plot areas: individual vehicle cost and team cost plots. The individual vehicle cost graphs are controllable by the user while the team cost graph simply displays the sum of the individual costs. Similar to the temporal control interface, users can drag any of three individual vehicle cost graphs using a mouse and release at any time when it reaches a desirable cost.

4.3.4 Coordination Variable/Function Interface

The coordination variable/function control allows users access to all three variables (temporal, spatial, and cost) in one display. In this method, users can utilize all three information types to find an optimal solution during the decision-making process. Figure 4.2(b) is a representation of the coordination function and variable. Each colored line (red, green, and blue) consists of piecewise lines indicating different spatial information.
Figure 4.3: Example of spatial control interface used in testing. As the user clicks spatial control buttons, other dependent variables change accordingly.

Figure 4.4: Example of cost control interface. The individual vehicle cost graphs are controllable by users. The outer boxes on each cost graph indicate the selectable range of vehicle costs.
In other words, this graph displays not only a relationship between time and cost, but it also represents spatial information corresponding to time. This graph enables users to find out a cost of a certain vehicle at any given time as well as vehicle’s path information. Similar to the temporal control interface, the vertical line indicates a desired ETA. By adjusting the location of the vertical bar, users can determine ETAs and paths of vehicles simultaneously.

4.4 Testing Method

A number of parameters can be measured among the four interfaces. Determining cause and effect relationships among these parameters and their interactions was done by statistical analysis. A full factorial design of experiments (DOE) was carefully performed to observe the quality and efficiency of each interface. To evaluate the quality and effectiveness of each interface quantitatively, three metrics (response time, workload, and quality of decision) were measured.

The response time measures the length of time it takes the user to determine a new cooperative timing solution when a roadblock appears during a simulation. The workload measures how much work is needed to find a solution. In order to increase the accuracy of the workload metric, two different measuring methods are used: measuring the total number of clicks in the interface, and evaluating the National Aeronautics and Space Administration (NASA) task load index (TLX) [53].

NASA TLX, shown in Figure 4.5, is designed to precisely test the efficiency of interfaces by measuring their workload according to six different criteria such as mental demand, physical demand, temporal demand, user performance, interface effectiveness, and user frustration. By comparing the two forms of data collected, a measure of the workload required by the four interfaces was determined. The quality of the decision is an average value of the operational costs that result from user choices during each test. These costs are gathered and analyzed from each interface test to measure the quality of decisions that users make based on the interface.

Besides four different interface designs, two levels of map settings (shown in Figure 4.6) and three levels of constraints (unconstrained, constrained path, and constrained cost) are used as qualitative factors for the multilevel full factorial DOE to measure these
Figure 4.5: Example of NASA TLX used to measure the workload of each interface.

Figure 4.6: Two levels of map settings used in testing. (a) Sparse field. (b) Dense field.
three metrics. Unconstrained cases allow users to make any decision as desired. Figure 4.7 illustrates an example of the constrained path case. In this case, some path choices are restricted and the user must choose paths for vehicles from the permitted side. Similar to constrained path cases, constrained cost cases make user choose solutions which are restricted within narrow cost limits. To emphasize the relative importance of vehicles, a specific vehicle cost ordering, in the form of the constrained cost, is given to users to follow when they make a decision. Table 4.2 illustrates a sample running order of the test. A total of 30 subjects (26 males and 4 females), whose ages ranged from 20 to 31 years, were used to collect the data in this thesis. Each test subject participated in 24 different tasks: a combination of four interfaces, two map settings, and three different constraints. The average time that each test subject spent to complete the testing was 74 minutes with standard deviation of 13 minutes. In every replication, the actual running order is randomized to increase confidence in the result.

![Permitted side vs. Prohibited side](image.png)

Figure 4.7: Example of the constrained path case. Users are limited to choose paths from the permitted side only.
### Table 4.2: Design of experiments, an example of a running order

<table>
<thead>
<tr>
<th>Running Order</th>
<th>Interface</th>
<th>Obstacles</th>
<th>Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Temporal control only</td>
<td>Less obstacles</td>
<td>Unconstrained</td>
</tr>
<tr>
<td>2</td>
<td>Temporal control only</td>
<td>Less obstacles</td>
<td>Constrained cost</td>
</tr>
<tr>
<td>3</td>
<td>Temporal control only</td>
<td>Less obstacles</td>
<td>Constrained path</td>
</tr>
<tr>
<td>4</td>
<td>Temporal control only</td>
<td>More obstacles</td>
<td>Unconstrained</td>
</tr>
<tr>
<td>5</td>
<td>Temporal control only</td>
<td>More obstacles</td>
<td>Constrained cost</td>
</tr>
<tr>
<td>6</td>
<td>Temporal control only</td>
<td>More obstacles</td>
<td>Constrained path</td>
</tr>
<tr>
<td>7</td>
<td>Spatial control only</td>
<td>Less obstacles</td>
<td>Unconstrained</td>
</tr>
<tr>
<td>8</td>
<td>Spatial control only</td>
<td>Less obstacles</td>
<td>Constrained cost</td>
</tr>
<tr>
<td>9</td>
<td>Spatial control only</td>
<td>Less obstacles</td>
<td>Constrained path</td>
</tr>
<tr>
<td>10</td>
<td>Spatial control only</td>
<td>More obstacles</td>
<td>Unconstrained</td>
</tr>
<tr>
<td>11</td>
<td>Spatial control only</td>
<td>More obstacles</td>
<td>Constrained cost</td>
</tr>
<tr>
<td>12</td>
<td>Spatial control only</td>
<td>More obstacles</td>
<td>Constrained path</td>
</tr>
<tr>
<td>13</td>
<td>Cost control only</td>
<td>Less obstacles</td>
<td>Unconstrained</td>
</tr>
<tr>
<td>14</td>
<td>Cost control only</td>
<td>Less obstacles</td>
<td>Constrained cost</td>
</tr>
<tr>
<td>15</td>
<td>Cost control only</td>
<td>Less obstacles</td>
<td>Constrained path</td>
</tr>
<tr>
<td>16</td>
<td>Cost control only</td>
<td>More obstacles</td>
<td>Unconstrained</td>
</tr>
<tr>
<td>17</td>
<td>Cost control only</td>
<td>More obstacles</td>
<td>Constrained cost</td>
</tr>
<tr>
<td>18</td>
<td>Cost control only</td>
<td>More obstacles</td>
<td>Constrained path</td>
</tr>
<tr>
<td>19</td>
<td>CVF control only</td>
<td>Less obstacles</td>
<td>Unconstrained</td>
</tr>
<tr>
<td>20</td>
<td>CVF control only</td>
<td>Less obstacles</td>
<td>Constrained cost</td>
</tr>
<tr>
<td>21</td>
<td>CVF control only</td>
<td>Less obstacles</td>
<td>Constrained path</td>
</tr>
<tr>
<td>22</td>
<td>CVF control only</td>
<td>More obstacles</td>
<td>Unconstrained</td>
</tr>
<tr>
<td>23</td>
<td>CVF control only</td>
<td>More obstacles</td>
<td>Constrained cost</td>
</tr>
<tr>
<td>24</td>
<td>CVF control only</td>
<td>More obstacles</td>
<td>Constrained path</td>
</tr>
</tbody>
</table>

CVF: Coordination variables and functions

### 4.5 Results and Discussion

To analyze the collected data, an analysis of variance (ANOVA) test was performed using the Statistical Analysis System (SAS) software package. The ANOVA test was designed to determine which factors and interactions were significant for the multilevel factorial DOE. In this research, repeated measures analysis with Satterthwaite’s approximation [54] was used to increase the accuracy of the results. Repeated measures analysis is a type of ANOVA in which multiple measurements are taken from each test subject. By correctly
considering each type of variation in results from different test subjects and different responses in the same situation from the same test subject, the repeated measures analysis improves the confidence of the results.

Satterthwaite’s approximation provides a better estimate of the degrees of freedom needed in a statistical test. Degrees of freedom determine the type of probability used in statistical analysis. A better estimate of the degrees of freedom results in a better estimate of the variance, thus this approximation provides more accurate results. The results are analyzed in three categories: response time, workload, and quality of decision.

4.5.1 Response Time

The first category analyzed is response time of users. The response time represents total elapsed time measured immediately after a problem is given to the user until the user solves and submits an answer. Table 4.3, shown below, presents the result of the ANOVA test for response time. According to the analysis with 90% confidence level, all effects but obstacles and interface variations affect the response time. The data suggests that the variation of interface is marginally significant and does not influence the response time significantly.

<table>
<thead>
<tr>
<th>Effect</th>
<th>NDF\textsuperscript{a}</th>
<th>DDF\textsuperscript{b}</th>
<th>F value</th>
<th>Prob. &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
<td>3</td>
<td>696</td>
<td>2.12</td>
<td>0.1041</td>
</tr>
<tr>
<td>Obstacles</td>
<td>1</td>
<td>696</td>
<td>1.47</td>
<td>0.2346</td>
</tr>
<tr>
<td>Constraint</td>
<td>2</td>
<td>696</td>
<td>6.73</td>
<td>0.0023</td>
</tr>
<tr>
<td>Interface x Obstacles</td>
<td>3</td>
<td>696</td>
<td>3.03</td>
<td>0.0337</td>
</tr>
<tr>
<td>Interface x Constraint\textsuperscript{**}</td>
<td>6</td>
<td>696</td>
<td>4.25</td>
<td>0.0005</td>
</tr>
<tr>
<td>Obstacles x Constraint</td>
<td>2</td>
<td>696</td>
<td>3.79</td>
<td>0.0283</td>
</tr>
<tr>
<td>Interface x Obstacles x Constraint\textsuperscript{**}</td>
<td>6</td>
<td>696</td>
<td>4.21</td>
<td>0.0006</td>
</tr>
</tbody>
</table>

\textsuperscript{a} Numerator degrees of freedom
\textsuperscript{b} Denominator degrees of freedom
\textsuperscript{**} Prob. < 0.001
Figure 4.8 illustrates the mean response time measured using various interfaces. This figure suggests that there is no significant evidence that one of the four interfaces is superior to the others. In other words, there is no significant difference in response time based on the interface. As the error bars indicates, the variation within data is large since the test subjects respond differently. However, it is necessary to investigate the response time based on the variation of constraints because Table 4.3, shown above, suggests that variations in the constraints are the most influencing factors with a 99.9% confident level.

Figure 4.9 displays four categories of response time: three different constraint options and the combined overall response time. The mean response time does not indicate any significant difference among different interfaces; however, as is observed by examining different constraint options, Figure 4.9 suggests otherwise. In general, response times of the spatial-only control interface are longer. However, in the constrained path case, Figure 4.9 indicates that the spatial-only control interface requires the least time. Consequently,
we can conclude that path constraints affect the overall response time more than the choice of user interface.

### 4.5.2 Workload

User workload in this research is defined as the total number of adjustments made by a user to finalize a solution. To increase the credibility of the test, two different methods were implemented to collect two types of workload measurements. The first measurement type is the total number of clicks by each user. During the simulations, the total number of clicks made by each user was collected for all 24 cases.

Table 4.4 represents the relationship between the total number of adjustments and different factors. This table suggests that the effect of the interface alone and the combined effects of the interface with other factors significantly affect the total number of adjustments, except for the three-way interaction (interface x obstacles x constraints). As shown in Figure 4.10(a), the number of adjustments noticeably changes based on the interface.
Table 4.4: Repeated measures ANOVA: Adjustment

<table>
<thead>
<tr>
<th>Effect</th>
<th>NDF(^a)</th>
<th>DDF(^b)</th>
<th>F value</th>
<th>Prob. &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface**</td>
<td>3</td>
<td>696</td>
<td>173.00</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Obstacles</td>
<td>1</td>
<td>696</td>
<td>0.30</td>
<td>0.5882</td>
</tr>
<tr>
<td>Constraint</td>
<td>2</td>
<td>696</td>
<td>0.69</td>
<td>0.5067</td>
</tr>
<tr>
<td>Interface x Obstacles**</td>
<td>3</td>
<td>696</td>
<td>18.34</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Interface x Constraint**</td>
<td>6</td>
<td>696</td>
<td>4.79</td>
<td>0.0001</td>
</tr>
<tr>
<td>Obstacles x Constraint</td>
<td>2</td>
<td>696</td>
<td>3.01</td>
<td>0.0573</td>
</tr>
<tr>
<td>Interface x Obstacles x Constraint</td>
<td>6</td>
<td>696</td>
<td>1.38</td>
<td>0.2266</td>
</tr>
</tbody>
</table>

\(^a\) Numerator degrees of freedom
\(^b\) Denominator degrees of freedom

** Prob. \leq 0.0001

Figure 4.10: Both figures indicate that the coordination variable/function method requires the lowest workload. (a) Workload measured by total number of adjustments during operation. (b) Workload measured by NASA TLX test.
used. Note that the *cost-only interface* and the *coordination variable and function interface* require minimal adjustments by users to determine the solution, while the spatial-only control requires almost four times more adjustments to achieve an acceptable solution.

In conjunction with the total number of adjustments, a NASA TLX test is conducted at the end of each test as a second measure of workload. As a user finishes each test, a survey is provided and the user rates the interface just used on a scale of one to 100 based on the given categories. The survey consists of six categories: temporal demand, mental demand, physical demand, effort required, performance, and frustration. Figure 4.10(b) illustrates the result of the NASA TLX test with confidence intervals. The result suggests that the coordination variables and functions interface has the minimum workload while the spatial-only control interface demands the most work.

### 4.5.3 Quality of Decision

The quality of decision is another way to measure the efficiency of interfaces. Under the same given conditions, users try to find the minimum operational cost of the mission using the four different interfaces. Table 4.5 shows correlations of operational costs among the different factors. Note that all results of this ANOVA test stay within 99.9% confidence levels; in other words, all factors both individually and combined significantly affect the result. At the same time, it suggests that the interface is the most influential factor in determining the operational cost.

Figure 4.11 illustrates the result of the cost analysis. As the error bars indicate, most users are able to achieve the mission objective, which is to minimize the operational cost. However, it is important to notice that the cost of the spatial interface is much higher than those of the three other interfaces. This implies that there might be a disadvantage to using the spatial-only control interface. To gain greater insight, a second analysis was performed.

To find more accurate relations among factors, a new ANOVA test was performed after excluding all data from the spatial-only control interface. Table 4.6 displays the result of the new ANOVA test. The results of this new test are quite different from that of the previous test (see Table 4.5). This new analysis suggests that two-way interactions of
Table 4.5: Repeated measures ANOVA: Quality of the operational team cost

<table>
<thead>
<tr>
<th>Effect</th>
<th>NDF&lt;sup&gt;a&lt;/sup&gt;</th>
<th>DDF&lt;sup&gt;b&lt;/sup&gt;</th>
<th>F value</th>
<th>Prob. &gt; F&lt;sup&gt;2&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
<td>3</td>
<td>696</td>
<td>522.36</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Obstacles</td>
<td>1</td>
<td>696</td>
<td>58.50</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Constraint</td>
<td>2</td>
<td>696</td>
<td>26.16</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Interface x Obstacles</td>
<td>3</td>
<td>696</td>
<td>135.27</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Interface x Constraint</td>
<td>6</td>
<td>696</td>
<td>8.32</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Obstacles x Constraint</td>
<td>2</td>
<td>696</td>
<td>37.27</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Interface x Obstacles x Constraint</td>
<td>6</td>
<td>696</td>
<td>6.38</td>
<td>&lt;.0001</td>
</tr>
</tbody>
</table>

<sup>a</sup> Numerator degrees of freedom  
<sup>b</sup> Denominator degrees of freedom

Figure 4.11: Mean operational team cost of interfaces.
obstacles are the most influential factors for determining the operational cost with 99.9% confidence levels. This significant new finding suggests that environmental factors (e.g., the number of obstacles combined with the type of constraint) are the most influential elements in determining the global minimum value of the operational cost.

Table 4.6: Repeated measures ANOVA: Quality of the operational team cost. The data is obtained using without spatial interface.

<table>
<thead>
<tr>
<th>Effect</th>
<th>NDF&lt;sup&gt;a&lt;/sup&gt;</th>
<th>DDF&lt;sup&gt;b&lt;/sup&gt;</th>
<th>F value</th>
<th>Prob. &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
<td>2</td>
<td>522</td>
<td>0.72</td>
<td>0.4864</td>
</tr>
<tr>
<td>Obstacles</td>
<td>1</td>
<td>522</td>
<td>12.22</td>
<td>0.0005</td>
</tr>
<tr>
<td>Constraint</td>
<td>2</td>
<td>522</td>
<td>0.54</td>
<td>0.5852</td>
</tr>
<tr>
<td>Interface x Obstacles</td>
<td>2</td>
<td>522</td>
<td>44.21</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Interface x Constraint</td>
<td>4</td>
<td>522</td>
<td>2.79</td>
<td>0.0257</td>
</tr>
<tr>
<td>Obstacles x Constraint</td>
<td>2</td>
<td>522</td>
<td>55.99</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Interface x Obstacles x Constraint</td>
<td>4</td>
<td>522</td>
<td>0.61</td>
<td>0.6573</td>
</tr>
</tbody>
</table>

<sup>a</sup> Numerator degrees of freedom

<sup>b</sup> Denominator degrees of freedom

4.5.4 Discussion of Results

The results from the comparison of four interfaces allows the efficiency and workload at each interface to be determined. The results suggests that the variation of interface does not significantly affect to the response time and the quality of decision except for the spatial only interface. In addition, the results indicate that the outcomes of coordination variable/function and cost interfaces are slightly better than others at the quality of decision. However, the effect of the interface alone and the effects of two way interaction between interface and other factors significantly affect the workload. According to the results, the coordination variable and function control interface has the least amount of workload. In addition, the result suggests that the spatial control interface is difficult to use and consequently performed poorly overall. The next chapter will discuss the proposed graphical user interface which is designed based on the results obtained in this chapter.
Chapter 5

Proposed Graphical User Interface

5.1 Features of Proposed Graphical User Interface

The results obtained in Chapter 4 provide insights into the types of interfaces most useful for multi-vehicle control by a single user. Designing the proposed graphical user interface (GUI) in this chapter is based on the results acquired in Chapter 4. The results suggest that using the cost control and the coordination variable and function control interface can reduce the amount of workload. The spatial control interface has known disadvantages such as the highest workload and the worst quality of decision though it has an advantage in the response time with the constrained path cases. In addition, the coordination variable/function control interface has the same feature set used in the temporal control interface. Hence, the temporal control and spatial control interfaces are excluded in the proposed GUI.

The proposed GUI shown in Figure 5.1 consists of four distinct zones: coordination variable/function interface (zone 1), spatial map display (zone 2), cost interface (zone 3), and variable option selectors (zone 4). Each zone presents different information that helps users to determine a final timing decision. In addition to these four zones, a cluster of executable buttons below the cost interface allows a user to operate the GUI.

5.1.1 Zone 1: Coordination Variable/Function Interface

The coordination variable/function interface shown in Figure 5.2 is one of the key features of this GUI. The horizontal axis represents temporal information (e.g., time of arrival), while the vertical axis indicates the cost of each individual path. Each red, green, and
Figure 5.1: Environment of graphical user interface

Figure 5.2: Coordination variable/function interface. The horizontal axis represents the coordination variable (time of arrival), while the vertical axis represents the coordination function (cost of vehicle paths). (a) Example of the simultaneous arrival type. (b) Example of the tight sequential arrival type (c) Example of the loose sequential arrival type
blue color line on the graph indicates the cost of the given path of each vehicle. The default value for the ETA is set to the minimum overall team cost, which is the sum of each vehicle’s cost. In the case of simultaneous arrival, a single black vertical line indicates the ETA of all vehicles (see Figure 5.2(a)). In the case of sequential arrival, three separate vertical lines (red, green, and blue) will indicate each individual vehicle’s ETA (see Figure 5.2(b) and Figure 5.2(c)).

This feature also allows users to adjust the vehicles’ ETAs from the default suggested ETA based on a user’s temporal decisions. This feature accommodates human decisions into an automated system to maximize team effectiveness. In addition, users can see the interaction of each vehicle while adjusting ETA, which can increase the effectiveness of the decision and decrease decision-making time. To adjust the ETA of a vehicle, two methods are proposed. The first method is drag and drop. The user depresses and holds down the left button on the mouse while dragging the ETA line and then releases the mouse button at the desired location. The second method to adjust ETA is by clicking on the desired ETA. This may allow users to see the interaction among vehicles more quickly; however, it may not be as precise as dragging the line. Figure 5.3 illustrates changes in other variables as the ETA changes.

5.1.2 Zone 2: Spatial Map Display

The feature shown in Figure 5.4 displays spatial information such as a chosen path and its waypoints as well as a configuration of an urban environment including roadblocks and hazardous areas. Once all data is computed, this interface displays the spatial map including individual vehicle paths, waypoints, and urban environment variables based on the default ETA. As the ETA is changed by a user, the paths and waypoints also respond accordingly. The perimeter of a hazardous area is illustrated with a red line, while obstacles (e.g., buildings and roadblocks) are rendered with gray quadrilaterals. This spatial information supports users in making a more informed and effective decision.
Figure 5.3: Examples of changes in variables based on the ETA. As an ETA line moves, vehicle paths in the map window and the individual vehicle cost are changed to correspond the coordination function. (a) Default ETA: 339.5 seconds. (b) User defined ETA: 165.0 seconds. (c) User defined ETA: 550.0 seconds.
5.1.3 Zone 3: Cost Interface

Figure 5.5 shows zone 3 of the GUI, which allows the user to view and control the cost of the mission. The left window displays information on each of the three individual vehicle costs, while the window on the right side shows the total team cost information. The values in the cost interface can be adjusted in two ways. One way to adjust vehicle costs is by setting the team ETA. As the user changes the ETA, both individual vehicle costs and the total cost become dependent variables. The other method of manipulating cost information is to change it directly through the interface. The user may choose any one of the three vehicle’s cost graphs and simply drag the chosen cost graph. The ETA, the chosen paths, the remaining two vehicle costs and the team total cost will automatically respond to ensure that the arrival constraint is met. This feature is useful for determining ETA with cost restrictions on the individual vehicles. For instance, if the safety of a certain vehicle must not be compromised, this feature allows a user to directly control this vehicle’s cost through the user interface so that its safety can be assured. Meanwhile, the remaining variables (e.g., ETA, paths, other vehicle costs, and the team cost) are automatically adjusted according to the chosen vehicle cost. In this case, the minimum team cost

Figure 5.4: Example of the spatial map display. The hazardous area is highlighted with red.
Figure 5.5: Vehicle cost analysis windows. The left window represents an individual vehicle cost while the right window represents the cumulative team cost.

is not guaranteed, nonetheless it provides a user the freedom to choose and receive instant visual feedback on the consequences of that choice.

5.1.4 Zone 4: Variable Option Selectors

The options box (Zone 4), displayed in Figure 5.6, consists of five distinctive areas of different options: arrival type selector, timing options selector, vehicle order selector, autonomous mode selector, and parameter option selector. The arrival type and timing option selector allow a user to select a different arrival type and to specify the timing constraints. The sequential order selector is activated when either the tight or loose sequence mode is selected. This allows a user to choose any vehicle order as desired. The autonomous mode selector provides a user with the freedom to select the autonomy mode during execution. The parameter option selector enables a user to choose one of many preset parameters (e.g., maps).

Arrival Type and Timing Option Selector

There are a total of three different timing options available. The first step in operating the GUI is to choose one of three arrival types (e.g., simultaneous, tight sequential, or
loose sequential arrival) shown in Figure 5.6 (a). Depending on the selection of the arrival type, appropriate features of the timing option selector will be activated. If the user chooses one of the two sequential arrival options, time editor boxes are activated, so the user can enter any desired timing constraint values. The values default to zero. To set any timing constraints (e.g., $T_{12}$, $T_{23}$, $\tau_1$, $\tau_2$, or $\tau_3$), the default values must be adjusted prior to the execution of the program by clicking the executable button at the bottom right of the GUI.

Figure 5.6 (b) shows the feature of the timing option selector. $T_{12}$ and $T_{23}$ indicate the timing offset between vehicle 1 and 2, and vehicle 2 and 3 respectively. $\tau_1$, $\tau_2$, and $\tau_3$ are timing slack variables which indicate that the execution is successful as long as each vehicle arrives within the windows defined by the timing slack variables (loose sequential arrive only). The default setting for this timing option is a simultaneous arrival. If this mode is selected, all timing constraint variables (e.g., $T_{12}$, $T_{23}$, $\tau_1$, $\tau_2$, and $\tau_3$) default to zero.
Vehicle Order Selector

When the user chooses either of the two sequential arrival modes (tight or loose sequence), the window shown in Figure 5.6 (c) will automatically be activated. Since this GUI limits the total number of vehicles to three, six possible permutations exist. However, all six of the arrival orders may not be available due to user-specified timing constraints. In this case, any unavailable ordering will be disabled and the GUI will automatically switch to the default value (minimum team cost). After the execution of the GUI, this option selector also allows the user to freely explore any available arrival sequential option.

Autonomy Option Selector

In this research, four specific levels (1, 3, 6, and 10) of autonomy from among the ten levels described in Table 2.1 have been investigated, and two levels of autonomy (3 and 10) were applied to the proposed GUI. The feature shown in Figure 5.6 (d) allows users to choose one of two autonomy modes: fully autonomous mode and semi-autonomous mode. The default value of this autonomy level is semi-autonomous mode. If the semi-autonomous mode is selected, the GUI will wait for user input (i.e., user-defined ETA) after the execution of the GUI before the simulation starts. In the other case, the fully autonomous mode automatically returns the ETA based on system default values and performs a simulation by itself. In other words, the fully autonomous mode will skip user input and execute the simulation based on pre-coded default values. During the simulation, the user can confirm the expected ETAs, paths, and individual cost as well as team costs from the simulation window.

Parameter Option Selector

In this GUI, a user can choose one of many pre-determined scenarios using the parameter option selector shown in Figure 5.6 (e). A simple pop-up menu is provided to assist the user in selecting preset scenarios. Prior to the simulation, many simulation parameters such as the locations of obstacles, coordinates of targets and initial vehicle positions, the location and the radius of the hazardous zone, and maneuverable vehicle
velocity range for each vehicle must be determined. When the user selects one of these scenarios, the chosen parameters are loaded into the program.

5.2 Simulations

To demonstrate a cooperative timing problem in an urban environment, several different simulation scenarios are investigated and a few are chosen for testing. Among these chosen scenarios, some assumptions are made. The list below describes all assumptions made for chosen simulation scenarios.

- All vehicles must avoid obstacles while maneuvering.
- There is one hazardous area (e.g., sniping zone, toxic waste spilled) in each scenario.
- If a vehicle enters a hazardous area, the vehicle must change its current velocity to the maximum velocity for the rest of its path.
- All vehicle positions and velocities are locally controlled.
- Collisions among vehicles are ignored.
- Vehicle paths must be contained within the map.

5.2.1 Simulation Setup

All scenarios are constructed inside a 1.5 km by 1.5 km square area. This particular area emulates a typical urban area where roads and buildings are located. In this scenario, all roads represent a feasible passageway to move, while all buildings indicate obstacles (e.g., no-go zone). In addition, there is a hazardous zone in every scenario. The hazardous zone represents an immediate threat to all moving vehicles within its radius. For all simulations, the total number of vehicles is limited to three to limit the complexity of the problem while demonstrating the functionality of the GUI design. Prior to simulation, all parameters must be determined and input to the system. Table 5.1 describes different parameters used for the simulation.

Parameters are divided into three major categories: environmental parameters, vehicle parameters, and timing constraints. The number of obstacles and their location are
Table 5.1: Simulation parameters used in simulations.

### The types of parameters

- **Environmental parameters**
  - Obstacles: 49 various locations in the field
  - Hazardous area location and its radius (km): X=0.375, Y=0.583, Radius=0.417
  - Target locations:
    - Vehicle 1 (Red): X=1.417, Y=1.417
    - Vehicle 2 (Green): X=1.417, Y=0.833
    - Vehicle 3 (Blue): X=1.417, Y=0.167

- **Vehicle parameters**
  - Number of vehicles: 3
  - Vehicle velocity range (m/s):
    - Vehicle 1 (Red): 1.39 ~ 13.89
    - Vehicle 2 (Green): 1.39 ~ 13.89
    - Vehicle 3 (Blue): 1.39 ~ 13.89
  - Vehicle initial locations:
    - Vehicle 1 (Red): X=0.167, Y=0.583
    - Vehicle 2 (Green): X=0.167, Y=0.333
    - Vehicle 3 (Blue): X=0.167, Y=1.333

- **Timing constraints (second)**
  
<table>
<thead>
<tr>
<th>Timing constraint</th>
<th>S.A.</th>
<th>T.S.A.</th>
<th>L.S.A.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{12}$</td>
<td>0</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>$T_{23}$</td>
<td>0</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>$\tau_1$</td>
<td>0</td>
<td>0</td>
<td>60</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>0</td>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>0</td>
<td>0</td>
<td>70</td>
</tr>
</tbody>
</table>

  
  - S.A. Simultaneous Arrival
  - T.S.A. Tight Sequential Arrival
  - L.S.A. Loose Sequential Arrival
determined and set as environmental parameters. In addition, the location of the hazardous area and its radius as well as target destinations for vehicles must also be provided prior to the simulation. Depending on each scenario, multiple vehicles may approach the same destination.

After environmental parameters are set, vehicle parameters must be entered. Vehicle parameters contain information regarding the condition of each vehicle, including initial positions for each vehicle and vehicle velocity range for each vehicle. Initial positions of all three vehicles can be varied for each scenario. In some cases, two or all vehicles may start from the same initial location side by side. In other cases, all vehicles may start from a different location.

Timing constraints are the most important components of this simulation because all simulation results must satisfy these timing constraints. As mentioned in Sections 3.2.2 and 3.2.3, there are five different constraints in this simulation: $T_{12}$, $T_{23}$, $\tau_1$, $\tau_2$, and $\tau_3$. The default values for these constraints are 0, which represents the simultaneous arrival case. Once a user chooses one of three timing options (i.e., simultaneous arrival, tight sequence, and loose sequence), appropriate timing constraints should be provided. For instance, if a user chooses the tight sequence option, the $T_{12}$ and $T_{23}$ values must be entered while $\tau_1$, $\tau_2$, and $\tau_3$ remain as 0.

5.2.2 Simulation Results

Figure 5.7 presents a simulation result where the red, green, and blue octagons indicate three different UGVs. Each colored line with circles illustrates a vehicle trajectory along with waypoints, whereas the smaller dots along the vehicle path indicate every 20th time step interval. Note that these intervals demonstrate the vehicle velocity changes during the simulation. The green and blue UGVs, for instance, have trajectories containing a hazardous zone. The intervals before these two vehicles entered the hazardous zone are closer to each other than those during and after exiting the zone. This indicates that vehicles increased their velocity during simulations when their trajectories passed through the hazardous zone; otherwise, the vehicle velocity remains constant along the trajectory.
Simultaneous Arrival Timing Constraints

The result of a simultaneous arrival simulation is presented in Figure 5.8. In this simulation, all timing constraints remain at 0. The top figure shows the coordination functions. The desired team ETA indicated by a black vertical line was 339.5 seconds where the total team cost becomes minimized. Each line segment on the top figure indicates a different feasible trajectory for the vehicle (see Section 3.2.1). The bottom plot illustrates the distance remaining to the destination. A sudden slope change of blue and green lines indicates the velocity changes of vehicles as they entered the hazardous zone. The simulation result shows that all three vehicles arrived their targets on time as desired.

Sequential Arrival Timing Constraints

Similar to the simultaneous arrival type simulation, the following two simulations are performed in the same map setting. Timing intervals between vehicles, $T_{12}$ and $T_{23}$ are set as 100.0 seconds and 80.0 seconds respectively in both tight and loose sequential arrival
Figure 5.8: Result of simultaneous arrival simulation. Top figure is coordination variable/function of mission, whereas bottom figure represents the range to destination of vehicles.

Simulations. Slack variables, $\tau_1$, $\tau_2$, and $\tau_3$, of loose sequential arrival type are set as 60.0, 50.0, and 70.0 seconds accordingly, while those of tight sequential arrival type remain at 0 (see Table 5.1).

Figure 5.9 displays simulation results of sequential arrival constraints. Three vertical colored lines shown in Figure 5.9(a) indicate the desired ETA for each vehicle. Note that the order of the colored lines shows the vehicles’ order of arrival. The desired ETAs determined by the coordination functions of the green, blue, and red vehicles are 228.5, 328.5, and 408.5 seconds respectively. The bottom plot in Figure 5.9(a) shows that all three vehicles arrived on time as commanded.

Figure 5.9(b) presents the simulation result of the loose sequential arrival mode. Similar to the tight sequential arrival mode, the three colored lines indicate the desired
Figure 5.9: Results of sequential arrival simulations. Top figures are coordination variable/function representation of missions, whereas bottom figures represent the range to destinations of vehicles. (a) Result of tight sequential arrival mode. (b) Result of loose sequential arrival mode.

ETAs of vehicles while the three colored boxes indicate the slack variables. The desired ETAs of the green, blue, and red vehicles are 215.0, 319.5, and 449.5 seconds respectively. Note that the desired ETAs from the loose sequential arrival simulation are slightly different from those of the tight sequential arrival due to the flexibility of determining arrival times. Since slack variables allow the coordination functions to yield an optimal solution within a feasible time window, the ETAs of the green and blue vehicles in Figure 5.9(b) have changed from those in Figure 5.9(a) to minimize both the individual and the team cost. Note that the ETA of the green vehicle moves to the right where the vertex of the chosen trajectory is located as the ETA of blue vehicle moves to the left. The lower figure in Figure 5.9(b) also confirms that all vehicles satisfy their timing constraints.
5.3 Summary

In this chapter, the features and the performance of the proposed GUI were described. The proposed GUI introduced in this chapter was created based on the result of the experiments discussed in Chapter 4. A combination of coordination variable/function control and cost control is implemented within the GUI to help users determine effective decisions for timing-sensitive cooperative control missions involving multiple vehicle systems. The simulation results confirm that this GUI is capable of planning paths for a team of UGVs, while satisfying cooperative timing constraints.
Chapter 6

Conclusions and Future Work

The primary objective of this thesis was to develop a human interface to accomplish timing-sensitive cooperative control missions for systems involving multiple vehicles. This research investigated and implemented coordination variables and functions to plan trajectories for a team of unmanned ground vehicles (UGVs) while satisfying timing constraints. Four different control interfaces were tested and analyzed to evaluate the performance and quality of each type of interface. Based on the experimental results, a final proposed graphical user interface (GUI) was designed incorporating a combination of coordination variable/function control and cost control. The simulation results in Section 5.2.2 demonstrate that this GUI is capable of planning paths for vehicles based on cooperative timing constraints and enables users to make high quality decisions in deploying a group of vehicles.

Though this research limited the number of vehicles to three, it is possible to accommodate as many vehicles as desired, since all codes created to implement algorithms used in this research including trajectory generation and coordination variables and coordination functions are modular except for the GUI. The proposed GUI was designed as a prototype, and for simplicity the display was limited to three robots. However, principles of GUI design discovered in this research are applicable to the control of a variable number of robots. Specific modular GUI designs were not studied, but changing the proposed GUI to allow the control of more than three vehicles is simple if all design parameters such as the maximum number of vehicles and the size of GUI are known a priori.

As the total number of vehicles increases, methods of representing vehicle data and controls in the GUIs should be reconsidered. The current coordination variable and
coordination function control interface displays all vehicles’ operational costs. Though the display does not appear crowded yet, it is expected that the representation of coordination variables and coordination functions will become overcrowded as the number of vehicle increases. One possible solution is to display the total operational cost instead of individual vehicle cost throughout the coordination variable domain. Another possible solution is to add a vehicle cost display option that allows users to choose to observe a specific vehicle’s operational cost, rather than viewing all vehicle cost information at once.

Though this thesis work primarily focuses on the application of cooperative control to multiple vehicle systems, this work can be applied to other fields such as logistics and resource management. Since the coordination variable and coordination function technique can be used in applications that involve allocating resources in the time domain, it is suitable for utilization in many resource allocation problems.

There are several areas to consider for improvement though results from this research were successful. Investigation of the items listed below may result in improvements in the overall performance of the interface.

- The further investigation on path planning algorithms is suggested. In this research, the Voronoi graph method was used to generate sets of feasible path nodes. Recent research [55], however, indicates that the Voronoi graph method may not be reliable for all maps despite of its exceptional computation time. This investigation may result in the improvement the performance of the path generation and increase the reliability of the proposed interface.

- An improvement of the performance of the spatial-only control interface is necessary. The spatial-only control interface developed in this research was less effective than the other interfaces tested and its use resulted in high user workload. The results found in Section 4.5 confirm that the spatial-only control interface is challenging to use and show that it performed poorly according to some test criteria. To improve the performance of this interface, it would be necessary to develop a method of allowing users to control vehicle paths directly by selecting and manipulating individual waypoints without overcrowding the spatial representation.
• Results from extensive simulations suggest that a hardware realization can be accomplished. To do so, some modifications on existing implementation (e.g., providing trajectories of vehicles in real-time, integrating hardware into the control loop) are necessary. Applying featured principles to a team of unmanned rotating-wing aerial vehicles which have similar characteristics to UGVs should also be considered.
Bibliography


